Stepsize control for path tracking

Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese,
and Charles W. Wampler II

*Dedicated to our collaborator, mentor, and friend, Andrew Sommese,
by Bates, Hauenstein, and Wampler on the occasion of his sixtieth birthday.*

Abstract. When numerically tracking implicitly-defined paths, such as is required for homotopy continuation methods, efficiency and reliability are enhanced by using adaptive stepsize and adaptive multiprecision methods. Both efficiency and reliability can be further improved by adapting precision and stepsize simultaneously. This paper presents a strategy for adjusting precision and stepsize together to eliminate certain types of path failures that can occur when adapting the two quantities independently, while also reducing the computational effort expended per unit advance along the path.

This paper concerns path tracking algorithms for tracing out a one dimensional path defined implicitly by n equations in $n+1$ unknowns. In particular, we consider such algorithms when multiprecision calculations are available, that is, when the precision of the computations can be changed during the computation. We treat a common type of path tracker that uses an Euler predictor to step ahead along the tangent to the path and a Newton corrector to bring the predicted point closer to the path. The objective of this paper is to describe a heurstic for adjusting precision and stepsize together to reduce the computational cost of tracking the path while maintaining high reliability.

In fixed precision tracking, a trial-and-error approach to setting the stepsize is effective: shorten the step upon failure, and lengthen it upon repeated successes. If the level of precision is inadequate, the step may fail no matter how small the step is made, so the trial-and-error approach repeatedly shortens the stepsize until failure is declared due to lack of progress.

2000 *Mathematics Subject Classification.* Primary 65H10; Secondary 65H20, 65G50, 14Q99.

Bates was supported by Colorado State University and the Institute for Mathematics and Its Applications (IMA).

Hauenstein was supported by the Duncan Chair of the University of Notre Dame; the University of Notre Dame Center for Applied Mathematics; and NSF grants DMS-0410047 and DMS-0712910.

Sommese was supported by the Duncan Chair of the University of Notre Dame; and NSF grants DMS-0410047 and DMS-0712910.

Wampler was supported by NSF grants DMS-0410047 and DMS-0712910.

In the multiprecision setting, one has the flexibility of either changing the stepsize or changing the precision. In [**1**], the precision is set first, according to rules designed so that corrector steps computed by Newton's method have enough digits of accuracy to ensure convergence, assuming the initial guess is within the convergence zone. If the initial guess is not adequate, the corrector fails, and the algorithm responds by shortening the stepsize to try again. For a small enough step and a high enough precision, the prediction/correction cycle must succeed and the tracker advances along the path. One would hope that the only mode of failure is when the combination of high precision and small steps is so severe that one gives up due to the excessive burden on computational resources. However, in testing that algorithm, another mode of failure was discovered: for too large a stepsize, the predicted point can be far enough from the path that the rules set the precision too high that the algorithm fails before a decrease in stepsize is considered. In particular, this was observed in tracking paths defined by polynomials of high degree and occurred on the first step when the initial stepsize given by the user was too large.

One might fix this problem by trapping the precision overflow condition and responding with a decrease in the stepsize. While such an approach may work, we present a more effective alternative here.

Success of a step depends on having sufficient precision and a small enough stepsize, but increasing precision and decreasing the stepsize both inflate the computational cost. With exact arithmetic, the stepsize is limited by the requirement for the predicted point to stay within the convergence zone of the corrector. For each stepsize below this limit, there is some minimum precision necessary to converge within the allowed number of correction steps. This necessary level of precision monotonically decreases with stepsize, approaching in the limit the precision that just barely ensures that the final error given by Newton's method equals the desired accuracy. Somewhere between these two limits, there must be an optimal combination of stepsize and precision that minimizes the computational effort per unit advance along the path. However, spending too much computation to find this optimum is itself counterproductive. Accordingly, in this paper, we develop a heuristic for finding a near optimum. At the same time, we eliminate the mode of failure previously mentioned. These new rules have been implemented in our software package, Bertini [**2**].

1. The Main Idea

The Euler predictor and the Newton corrector can be summed up in a single relation, obtained by retaining only the linear terms in a Taylor series expansion about (z_i, t_i):

$$H_z(z_i, t_i)\Delta z + H_t(z_i, t_i)\Delta t = -H(z_i, t_i), \tag{1.1}$$

$$(z_{i+1}, t_{i+1}) = (z_i, t_i) + (\Delta z, \Delta t). \tag{1.2}$$

where $H_z = \frac{\partial H}{\partial z}$ and $H_t = \frac{\partial H}{\partial t}$. Let (z_0, t_0) be the current approximation of a point on the path along with its t value, so that $H(z_0, t_0) \approx 0$. Given stepsize Δt, (z_1, t_1) is the Euler prediction. Upon setting $\Delta t = 0$, (z_i, t_i) are Newton corrections at $t_i = t_1$ for $i \geq 2$.

We can write this sequence of iterates in such a way that it is the sequence generated by applying Newton's method to a system $f(z, t)$. This is useful because

the analysis of Newton's method in [**1**] now applies to both the prediction step and the corrector steps. This is the main idea of this paper: the analysis presented in [**1**] may be extended to a method that adaptively changes the stepsize and precision simultaneously. This new method increases the security of adaptive precision path tracking while simultaneously reducing the computational cost.

As above, let (z_0, t_0) be the current point approximately on the path and let s be the stepsize. Define $T = t_0 + s$ to be the target for t for the next point on the path and consider the augmented system

$$f(z,t) = \begin{pmatrix} H(z,t) \\ t - T \end{pmatrix} = 0. \tag{1.3}$$

Applying Newton's method to $f(z,t)$, we produce the sequence (z_{i+1}, t_{i+1}) by solving

$$\begin{bmatrix} H_z(z_i,t_i) & H_t(z_i,t_i) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta t \end{bmatrix} = - \begin{bmatrix} H(z_i,t_i) \\ t_i - T \end{bmatrix}, \tag{1.4}$$

$$(z_{i+1}, t_{i+1}) = (z_i, t_i) + (\Delta z, \Delta t). \tag{1.5}$$

It is easy to confirm that the sequence of iterates produced in this way are exactly the same as before: the first iterate is just the Euler prediction and subsequent ones are Newton corrections at $t = T$.

2. New Rules for Stepsize

2.1. Summary of adaptive precision method. The method of [**1**] depends on the enforcement of three rules that determine when precision should be changed to maintain the desired path-tracking accuracy. Since the present method extends that of [**1**], we follow the notation of that article. In particular, let P be the number of digits of precision, so that $u = 10^{-P}$ is the unit roundoff error. Further, let $10^{-\tau}$ be the accuracy to which we wish to track the path, let $\|\cdot\|$ denote a vector norm and its induced submultiplicative matrix norm, let $\|d\|$ be the most recent Newton residual, and let N be the maximum number of Newton iterations to perform.

Let $F(z) : \mathbb{C}^n \to \mathbb{C}^n$ be continuously differentiable, and let $J(z)$ denote its Jacobian matrix. Due to the nature of finite precision, there is error associated with evaluating the function and the Jacobian and also in solving a system of linear equations. Let $\psi(z,u)$ and $\phi(z,u)$ be the functions that account for the errors in evaluating $F(z)$ and $J(z)$, respectively, and let $\mathcal{E}$ be the constant that accounts for the growth in errors when solving a system of linear equations. Suppose that the error functions ψ and ϕ are of the form $\psi = \Psi u$ and $\phi = \Phi u$. Methods for approximating Ψ and Φ are given in [**1**]. Since the approximations used in the rules may underestimate the true values, extra digits, called safety digits in [**1**], are included. Let σ_1 and σ_2 denote the number of safety digits requested for the rules described below.

The first rule requires that the error perturbed Jacobian matrix needs to be nonsingular, namely

$$P > \sigma_1 + \log_{10}[\|J^{-1}\|\mathcal{E}(\|J\| + \Phi)]. \tag{**A**}$$

This rule is applied before entering the corrector loop when $\|d\|$ is not yet available.

After the first pass through the corrector, $\|d\|$ becomes available, and Eq. **A** is superceded by a more restrictive rule requiring that the corrector must converge

within N iterations. Define $D = \log_{10}\left[\|J^{-1}\|((2+\mathcal{E})\|J\| + \mathcal{E}\Phi) + 1\right]$ and suppose that there are $(N-i)$ Newton iterations remaining, the second rule is

(**B**) $$P > \sigma_1 + D + (\tau + \log_{10}\|d\|)/(N-i).$$

Roughly speaking, D is the number of digits lost to numerical error in computing corrections. The remaining digits of accuracy improve our approximate solution. If every iteration adds as many correct digits as the last term, $(\tau+\log_{10}\|d\|)/(N-i)$, then the final tolerance $10^{-\tau}$ will be reached within $(N-i)$ remaining iterations. Thus, setting P in accordance with Eq. **B**, gives enough precision to enable convergence.

The third rule requires that the final accuracy of the corrector be within the required tolerance, namely

(**C**) $$P > \sigma_2 + \tau + \log_{10}(\|J^{-1}\|\Psi + \|z\|).$$

Full details regarding the theoretical development of these rules, along with examples coming from the specific implementation of this method within Bertini [**2**] may be found in [**1**].

2.2. Combining adaptive precision and adaptive stepsize. In [**1**], Eq. **B** only applies to the corrector steps. By using the augmented system defined by Eq. 1.3, this rule applies to the Euler prediction step since it is just the initial Newton iteration. Including the prediction step as the first Newton iteration requires N to be increased by one. This setup now includes another parameter that we can vary besides the precision P, namely the stepsize s. The rest of this section describes how to adaptively change the precision P and the stepsize s together.

On the first iteration, the Euler prediction, $\|d\| = \|(\Delta z, \Delta t)\|$ is directly proportional to s, say $\|d\| = a|s|$. Accordingly, we may rewrite Eq. **B** for the first iteration as

$$P - \log_{10}|s|/N > \sigma_1 + D + (\tau + \log_{10} a)/N, \tag{4}$$

or, letting $|s| = 10^{-\xi}$, as

$$P + \xi/N > \sigma_1 + D + (\tau + \log_{10} a)/N. \tag{5}$$

There are two ways to satisfy this inequality: raise precision P or decrease the stepsize by raising ξ.

Suppose that $C(P)$ is the cost of computing N iterations in precision P. Then the cost per unit advance along the path is $C(P)/|s|$. By minimizing $C(P)/|s|$ subject to Eq. 5, P and s can be set optimally. One thing to remember is that this rule assumes that we are within the convergence zone of Newton's method. If not, additional precision will not be effective and the stepsize must be cut. Therefore, we must retain the previous algorithmic step of cutting the stepsize when convergence is not obtained within N Newton iterations.

2.3. Outline of the algorithm. The path tracking algorithm presented in this article assumes that the path tracker can reach the final value t_f without passing through an exact singularity on the path. When using homotopies to solve polynomial system, with probability one, no path passes exactly through a singularity, except possibly at the end. The example presented in Section 3.3 demonstrates that near singular conditions do occur during path tracking away from the end of the path.

The algorithm is outlined in the flowchart presented in Figure 1. Some key parts presented in the diagram require additional explanation. As in [**1**], there is a maximum number of steps and a maximum precision, $P_{\max}$, allowed. These limits prevent computational waste and guarantee the termination of the algorithm. "Setup $f(z,t)$" constructs $f(z,t)$ described by Eq. 1.3 combining the prediction step and correction steps. If the step in t would go beyond t_f, then the stepsize is adjusted to land exactly on t_f. The next box, labeled "Correct," computes a Newton correction. In the case that a correction cannot be computed due to singularity of the Jacobian matrix at the current precision, the algorithm branches to "Call *convergence_error*." Otherwise, the results of the correction are checked using safety rules defined by Eqs. **B** and **C**, invoking "Call *safety_error*," if either criteria is violated. If the safety rules are met, the algorithm checks for convergence to within $10^{-\tau}$, either proceeding to update or considering another correction cycle, as necessary. The remaining key parts, namely "Call *convergence_error*," "Call *safety_error*," and "Call *step_success*," are described below.

When the Newton iterations fail to converge, the *convergence_error* algorithm is executed. This algorithm decreases the stepsize as in the adaptive stepsize method. The new stepsize is then compared with the predetermined minimium stepsize for the given value of the precision, as discussed in [**1**], to avoid stalling on the path. The values used for the minimum stepsize, $\varepsilon(P)$, are given in Section 3.1.

Algorithm 1. *convergence_error*$(P_{in}, s_{in}, r; P_{out}, s_{out})$

Input:
- P_{in}: current precision.
- s_{in}: current stepsize.
- r: step adjustment factor, between 0 and 1, exclusive.

Output:
- P_{out}: new precision.
- s_{out}: new stepsize.

Algorithm:
Initialize $P_{out} := P_{in}$ and $s_{out} := r * s_{in}$.
While $s_{out} < \varepsilon(P_{out})$
    Increment P_{out} to the next available precision.

The *safety_error* and *step_success* algorithms both rely upon an algorithm, called *minimize_cost*, to adjust the precision and stepsize to avoid stalling on the path, satisfy Eq. 4, and reduce the cost. The precision selected by this algorithm is between the current precision and the maximum precision, inclusively, and the stepsize selected is not larger than the current stepsize.

Algorithm 2. *minimize_cost*$(P_{in}, s_{in}; P_{out}, s_{out})$

Input:
- P_{in}: current precision.
- s_{in}: current stepsize.

Output:
- P_{out}: new precision.
- s_{out}: new stepsize.

Algorithm:
Initialize $G := \emptyset$.
For each P from P_{in} to the maximum precision, P_{max},
    Compute $|s|$ that satisfies Eq. 4 as an equation,
    Append $(P, \min(|s|, |s_{in}|))$ to G if $\min(|s|, |s_{in}|) > \varepsilon(P)$.
If $G = \emptyset$,
    Set $P_{out} := P_{max} + 1$ and $s_{out} := s_{in}$.
Otherwise,
    Let $(P, |s|)$ be the pair that minimizes $C(P)/|s|$ among the choices in G,
    Set $P_{out} := P$ and $s_{out} = |s| * s_{in}/|s_{in}|$.

When either Eqs. **B** or **C** are not satisfied, the *safety_error* algorithm is executed. This algorithm increases the precision and decreases the stepsize to satisfy both criteria. If there is no change in precision, the correction step is merely rescaled to the new stepsize and the algorithm proceeds along branch *a*. On the other hand, if precision is changed, the algorithm takes branch *b* to circulate back to recompute the correction at higher precision.

Algorithm 3. *safety_error*$(P_{in}, s_{in}, d_{in}; P_{out}, s_{out}, d_{out})$

Input:
- P_{in}: current precision.
- s_{in}: current stepsize.
- d_{in}: current correction, $(\Delta z, \Delta t)$.

Output:
- P_{out}: new precision.
- s_{out}: new stepsize.
- d_{out}: new correction.

Algorithm:
Initialize $P_{out} := P_{in}$, $s_{out} := s_{in}$, and $d_{out} := d_{in}$.
While Eq. **C** is not satisfied,
    Increment P_{out} to the next available precision.
If this is the first pass through the correction loop,
    $(P_{out}, s_{out}) :=$ *minimize_cost*(P_{out}, s_{out}).
Otherwise, while Eq. **B** is not satisfied,
    Increment P_{out} to the next available precision.
If $P_{out} = P_{in}$
    Rescale the correction: $d_{out} = d_{in} * s_{out}/s_{in}$,
    Reset $f(z, t)$ to match the new stepsize, s_{out},
    Exit on branch *a*.
Otherwise,

Exit on branch b.

When a step is successfully completed, the *step_success* algorithm is executed. It tries to reduce the computational cost of the next step by decreasing precision or increasing the stepsize. However, the extent of these changes is limited by the necessity of conforming to Eqs. 4 and **C**. Moreover, to avoid acting too aggressively, which might waste computation by failing during the next correction cycle, the frequency of relaxation in precision or stepsize is limited. In particular, if there have been M successful steps in a row, the stepsize is allowed to increase, and if there have been L successful steps in a row, the precision is allowed to decrease. The number of successful steps in a row is reset back to zero after the number of consecutive successful steps reaches the larger of M and L.

Algorithm 4. *step_success*($numSuccess_{in}, P_{in}, s_{in}, r, M, L$; $numSuccess_{out}, P_{out}, s_{out}$)

Input:
- $numSuccess_{in}$: current number of successful steps in a row.
- P_{in}: current precision.
- s_{in}: current stepsize.
- r: step adjustment factor, between 0 and 1, exclusive.
- M: number of consecutive successful steps before an increase in stepsize.
- L: number of consecutive successful steps before a decrease in precision.

Output:
- $numSuccess_{out}$: new number of successful steps in a row.
- P_{out}: new precision.
- s_{out}: new stepsize.

Algorithm:
Initialize $numSuccess_{out} := numSuccess_{in} + 1$, $P_{out} := P_{in}$, and $s_{out} := s_{in}$.
If $numSuccess_{out} = M$
    $s_{out} := s_{out}/r$.
If $numSuccess_{out} = L$ and P_{out} is not the smallest available precision
    Set P_{out} to be the next lower available precision.
If $numSuccess_{out} = \max(M, L)$
    $numSuccess_{out} := 0$.
$(P_{out}, s_{out}) :=$ *minimize_cost*(P_{out}, s_{out}).

For completeness, we note that in the final line of algorithm *step_success*, we use the historical data from the most recent step along the path; that is, we use J from the last Newton iteration and $\|d\|$ from the first Newton iteration of the previous correction cycle.

3. Implementation Details and Computational Experiments

Adaptive multiprecision tracking with stepsize control is implemented in the software package Bertini [**2**]. Though the original adaptive precision method found in [**1**] was first implemented in Bertini beta, all examples discussed below were run

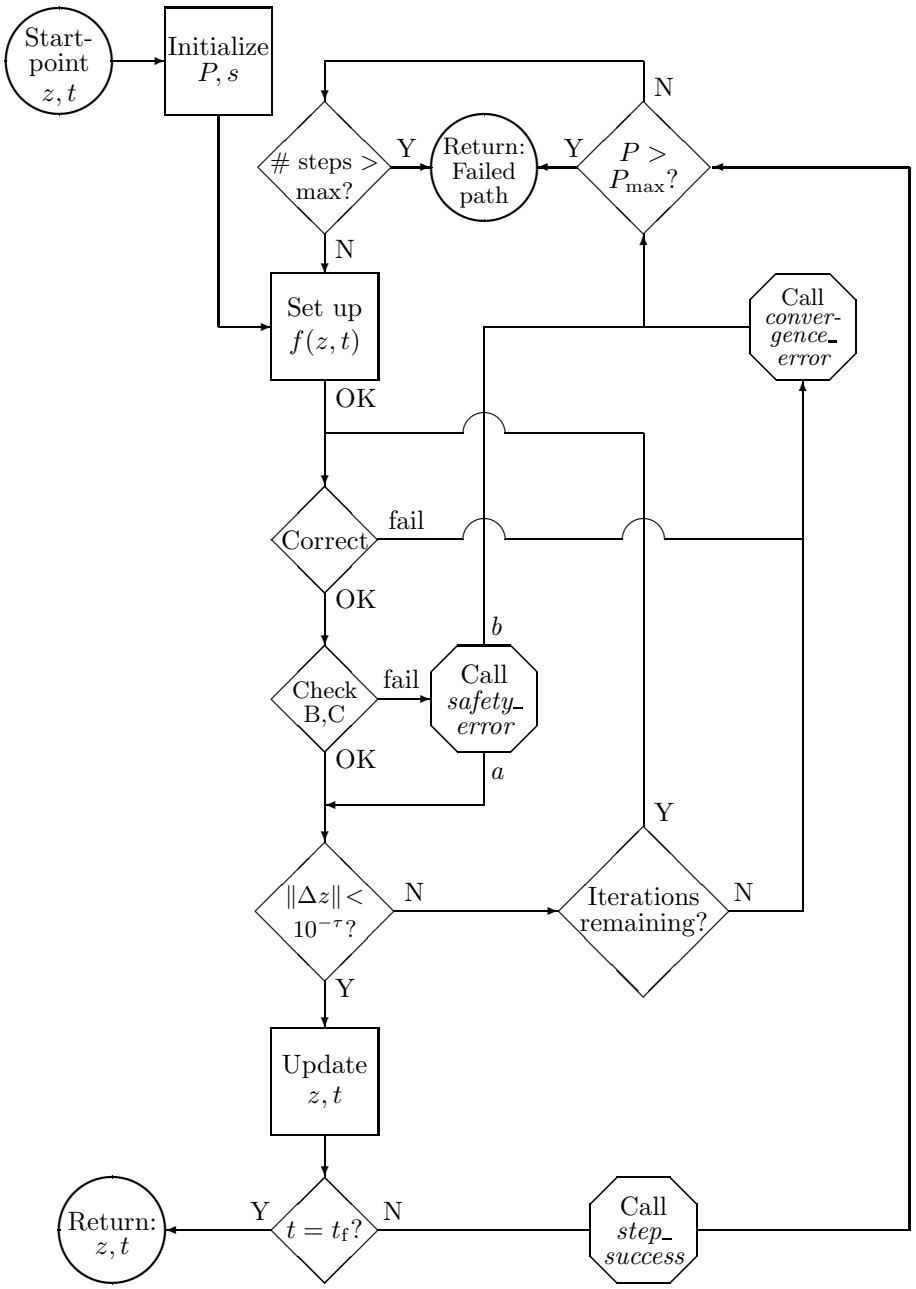

FIGURE 1. Adaptive precision path tracker with stepsize control. The subroutine calls are explained in the text.

on the common platform of Bertini v1.1 using an Opteron 250 processor running 64-bit Linux.

3.1. Implementation Details. Bertini uses MPFR for multiprecision arithmetic, which allows for precision to be changed in discrete packets of 32 bits. At

	IEEE double	MPFR				
bits of precision	52	64	96	128	160	192
P	16	19	28	38	48	57
$\varepsilon(P)$	10^{-14}	10^{-16}	10^{-25}	10^{-34}	10^{-43}	10^{-52}

TABLE 1. Values of $\varepsilon(P)$

minimum fixed precision		method of [**1**]	new method
96 bits	184.01	38.54	32.73

TABLE 2. Comparison for average time of 10 runs of the IPP system, in seconds.

various levels of precision in MPFR, we computed the time needed to perform common operations used in homotopy continuation, e.g. straight-line program evaluation, matrix multiplication and linear solving. These timings were compared with the time needed to perform these operations using IEEE double precision. Based on this data, we computed the cost function $C(P)$. With P in digits of precision, the cost function used in the following examples was

$$C(P) = \begin{cases} 1, & \text{if } P = 16 \text{ (i.e., corresponds to double precision);} \\ 10.35 + 0.13P, & \text{otherwise.} \end{cases}$$

As new versions of MPFR are released, this cost function will be recomputed.

In the examples below, $\sigma_1 = \sigma_2 = 1$, $M = 5$, $L = 10$, $N = 2$, $\tau = 6$, and the values for $\varepsilon(P)$ are presented in Table 1. The maximum precision, P_{max}, was set to 308 digits corresponding to 1024 bits of precision.

3.2. Comparing the methods. In Section 5.5 of [**1**], a polynomial system arising from the inverse kinematics problem for a general six-revolute serial-link robot [**3**] is considered. Utilizing the power series endgame with the same settings as in [**1**], Table 2 indicates the average time required to solve that system with fixed precision, the method of [**1**], and the new method of this paper.

The method described in this paper causes paths to be tracked using double precision longer than the method of [**1**] by decreasing the stepsize rather than automatically relying on the power (and cost) of higher precision when numerical difficulties are encountered. This resulted in a 15% improvement in speed for this example since double precision computation is so much less costly than multiple precision computation.

3.3. Near singular conditions. For the homotopies utilized, with probability one, the paths do not pass directly through a singularity on $(0, 1]$. Even though the Jacobian is still nonsingular, higher precision may be needed to reveal this. It is not known, *a priori*, how many paths travel near a singularity for a given homotopy.

To demonstrate that near singularity conditions do exist, consider the formulation of the nine-point path synthesis problem for four-bar linkages in [**4**]. Utilizing the 2-homogeneous structure and the two-fold symmetry, the homotopy consists of $143,360$ paths of which 4326 lead to nondegenerate solutions. The precision points were selected at random and the homotopy was created using random numbers.

During the tracking, 1184 of the total 143,360 paths (0.83%) needed to use precision higher than double to track past a near singularity before returning back to double precision. Moreover, 680 paths (0.47% of the total) needed to use at least 96 bits of precision to track past a near singularity before returning to double precision.

Figure 2 is a graph of the log of the condition number, precision and stepsize in relation to tracking parameter t for a typical path having a near singularity and requiring the use of at least 96 bits of precision before returning to double precision.

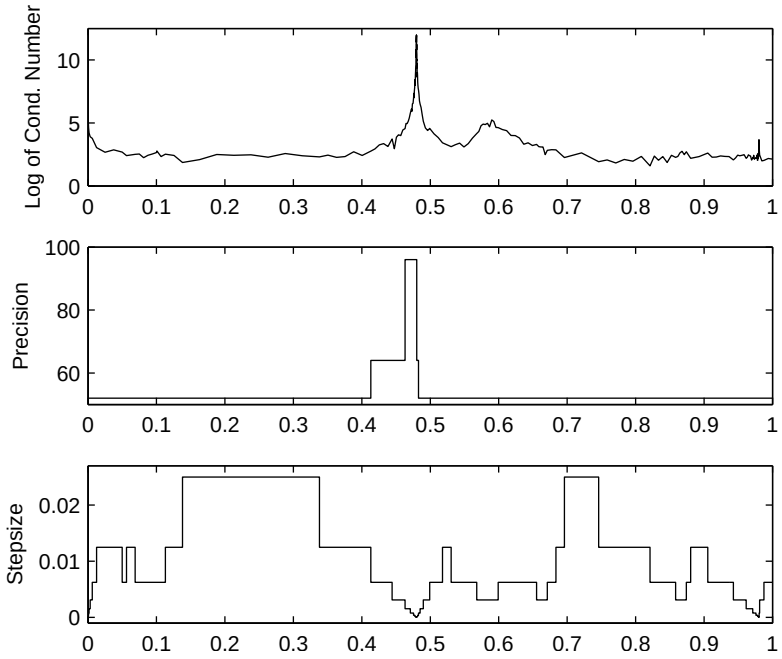

FIGURE 2. Graph of the log of the condition number, precision and stepsize against the tracking parameter t

4. Conclusion

Adaptive stepsize and adaptive multiprecision techniques may be employed to enhance the efficiency and especially the reliability of path-tracking methods, as with homotopy continuation. The performance of a tracking method can be improved by considering both adaptive procedures simultaneously, as opposed to handling them separately. This paper provides a strategy for adjusting precision and stepsize together, yielding both higher reliability and a reduction in computational burden. This technique is detailed in a flowchart, and the performance of the implementation of this technique in the Bertini software package is demonstrated with two examples.

References

1. D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, Adpative multiprecision path tracking, SIAM Journal on Numerical Analysis, 46(2): 722–746, 2008.

2. D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, Bertini: Software for Numerical Algebraic Geometry. Available at www.nd.edu/∼sommese/bertini.
3. A.P. Morgan and A.J. Sommese. Computing all solutions to polynomial systems using homotopy continuation. *Appl. Math. Comput.*, 24(2): 115–138, 1987. Errata: *Appl. Math. Comput.*, 51:209, 1992.
4. C.W. Wampler, A. Morgan, and A.J. Sommese, Complete solution of the nine-point path synthesis problem for four-bar linkages, *ASME Journal of Mechanical Design* 114(1): 153–159, 1992.

Department of Mathematics, Colorado State University, 101 Weber Building, Fort Collins, CO 80528
E-mail address: bates@math.colostate.edu, http://www.math.colostate.edu/∼bates

Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556
E-mail address: jhauenst@nd.edu, http://www.nd.edu/∼jhauenst

Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556
E-mail address: sommese@nd.edu, http://www.nd.edu/∼sommese

General Motors Research and Development, Mail Code 480-106-359, 30500 Mound Road, Warren, MI 48090
E-mail address: Charles.W.Wampler@gm.com, http://www.nd.edu/∼cwample1